Component-based Architecture
And
Modeling and Simulation

Simulation-Based Acquisition Conference Observations

- Dr Sega – OSD, Director Defense Research & Engineering
  - Platform-centric→ network centric
  - Common vision representation
    - Multiple function areas
    - Joint, interoperable, re-useable models
- Dr Dahmann – Scientific advisor to Director of Interoperability
  - System focus → mission focus
- Ms. Zimmerman – DMSO
  - Rapidly composable and scalable M&S
  - Strong Configuration Management focus
  - Build only what is needed
- Mr Lunceford – Director, Army M&S Office
  - M&S best practices still a mystery
  - ***Begin shift of M&S from craft to scientific/engineering discipline***

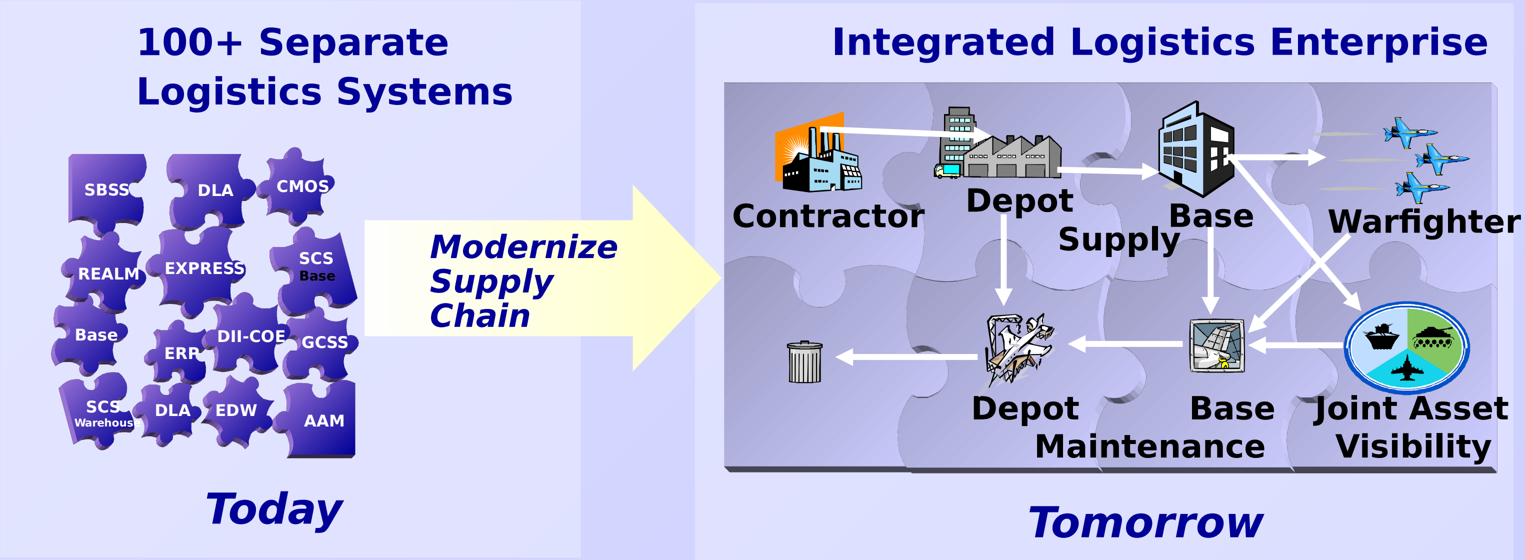
Today’s Goals

- Discuss
  - Describe AF component-based strategy and approach
    - The picture and the pieces
    - The story behind the design
  - Apply lessons learned to M&S
  - Components and the DoD transformation
    - Components are a way of thinking about systems and organizations
      - Not just IT
  - Challenges

- Generate ideas, discussions, and excitement

Background Projects

Component-based Supply Chain Modernization



100+ Separate Logistics Systems

Modernize Supply Chain

Integrated Logistics Enterprise

Today

Tomorrow

Air Force Logistics

- Collection of stovepiped systems
- Pieces do not connect
- Picture not complete

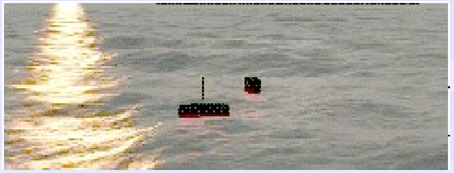
- Integrated picture across IL
- Complete connectivity
- Total Asset Visibility (TAV)
  - factory-to-foxhole

Background Projects

3D Component-based M&S Framework

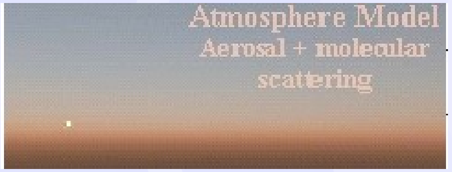
Component-based Models

Component-based Rendering



Arete Ocean Model
Ocean heights
Radiance function (point)

- Reflections
- Dynamic grid sampling
- Rendering



- Rendering
- Transmission

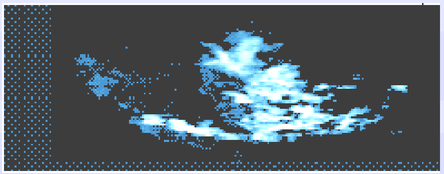
RenderWorld™ Visualization Framework

- **Transmission**

Sensor Motion
Object Interaction
Coordinate Transformations

- Scattering
- Volumetric Integration

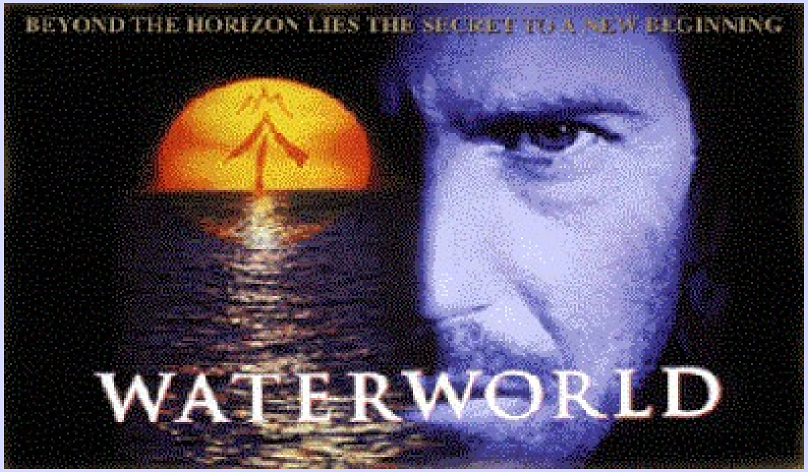
Arete Cloud Model
Liquid water content
Scattering phase
function

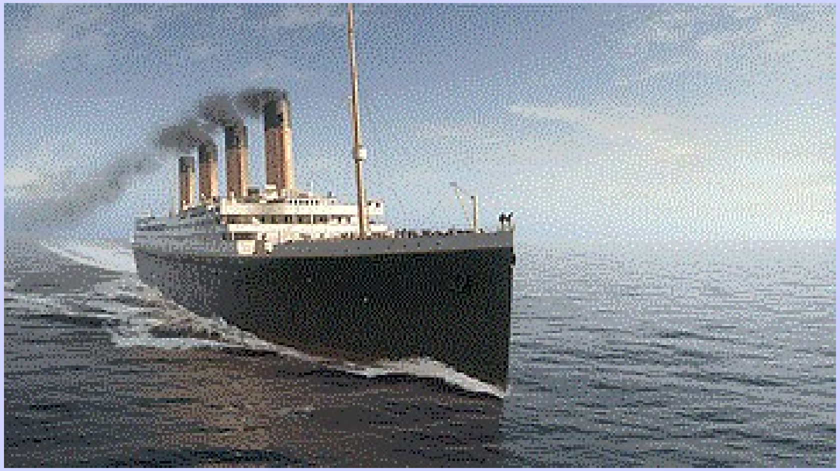


Serves as reference – not recommendation

Background Projects

Component Based Environment Simulator



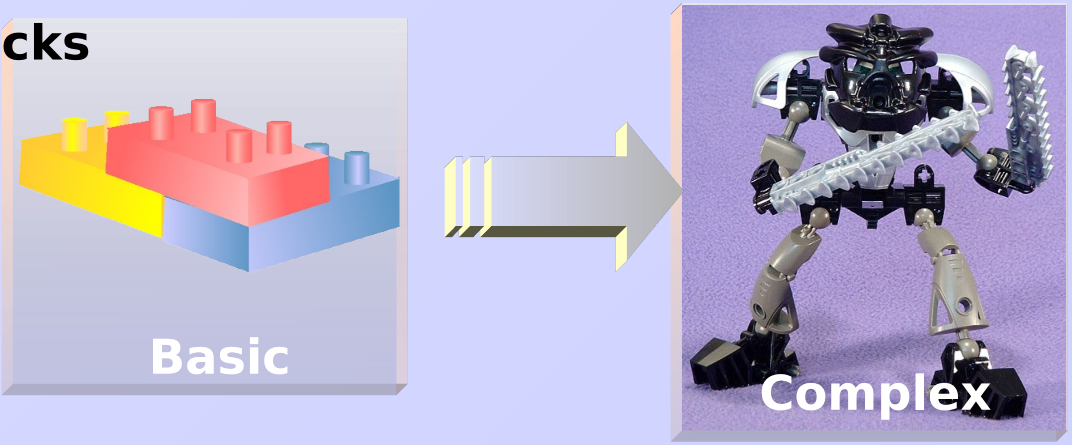


- Commercial Product (with DoD origins)
  - Limited success marketing it within DoD
    - M&S exists to support other programs – focus on solving those problems
    - Need mission-focused applications
- Component architecture makes it very flexible
  - Plug in model to numerous rendering packages

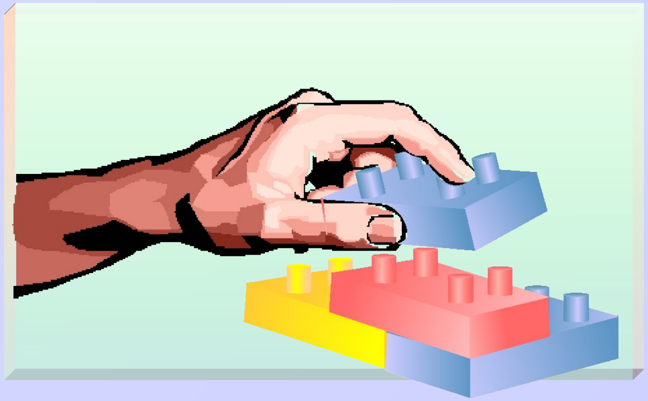
What is a component?

Software building blocks

- Structured interfaces
- Clear purpose
- Build complex apps



- **Examples**
  - Legos (complex and general purpose)
  - Dictionary in MS products

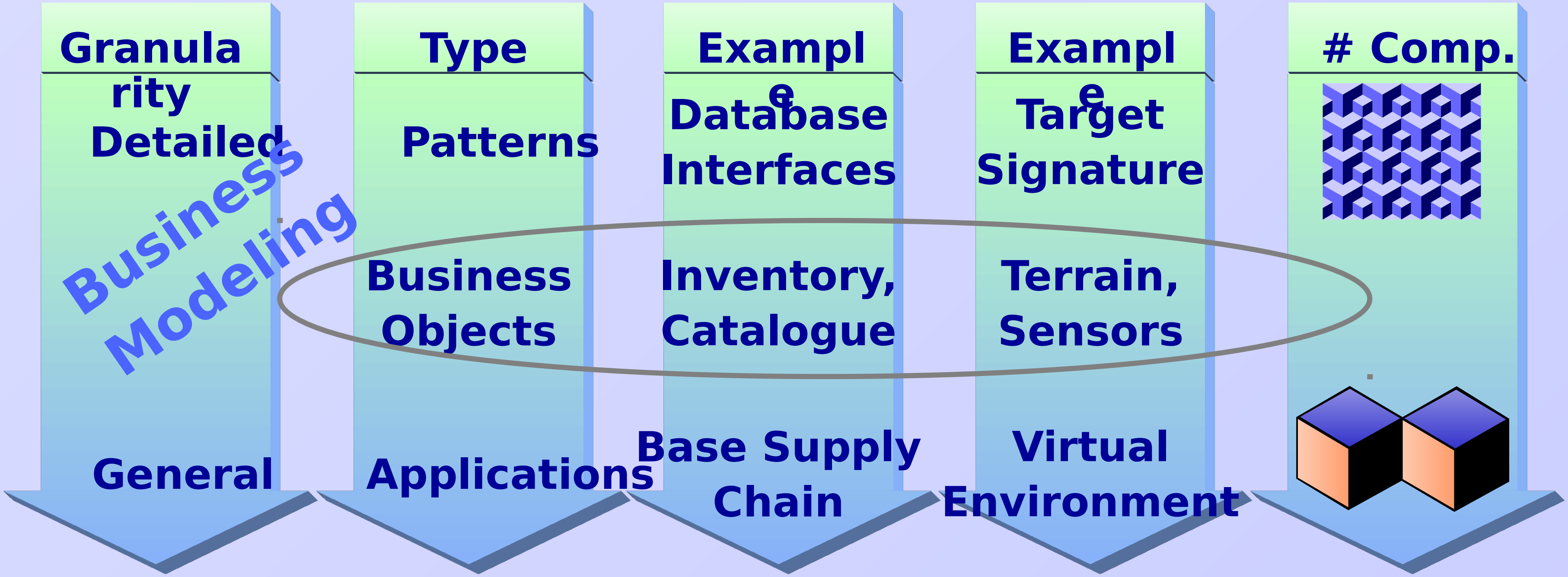


Component-based Development

- Connecting the pieces

Component Granularity

Components = unit of functionality



Granularity	Type	Example	Example	# Comp.
Detailed	Patterns	Database Interfaces	Target Signature	
	Business Objects	Inventory, Catalogue	Terrain, Sensors	
General	Applications	Base Supply Chain	Virtual Environment	

Business Modeling

Need **right** granularity: Too granular – exceedingly complex

Too general – limits reusability

Components Examples

Supply Chain Vocabulary

AF Logistics

- OrderManagement
- Inventory
- Catalog
- History
- Security
- Financial

M&S Vocabulary

Modeling and Simulation

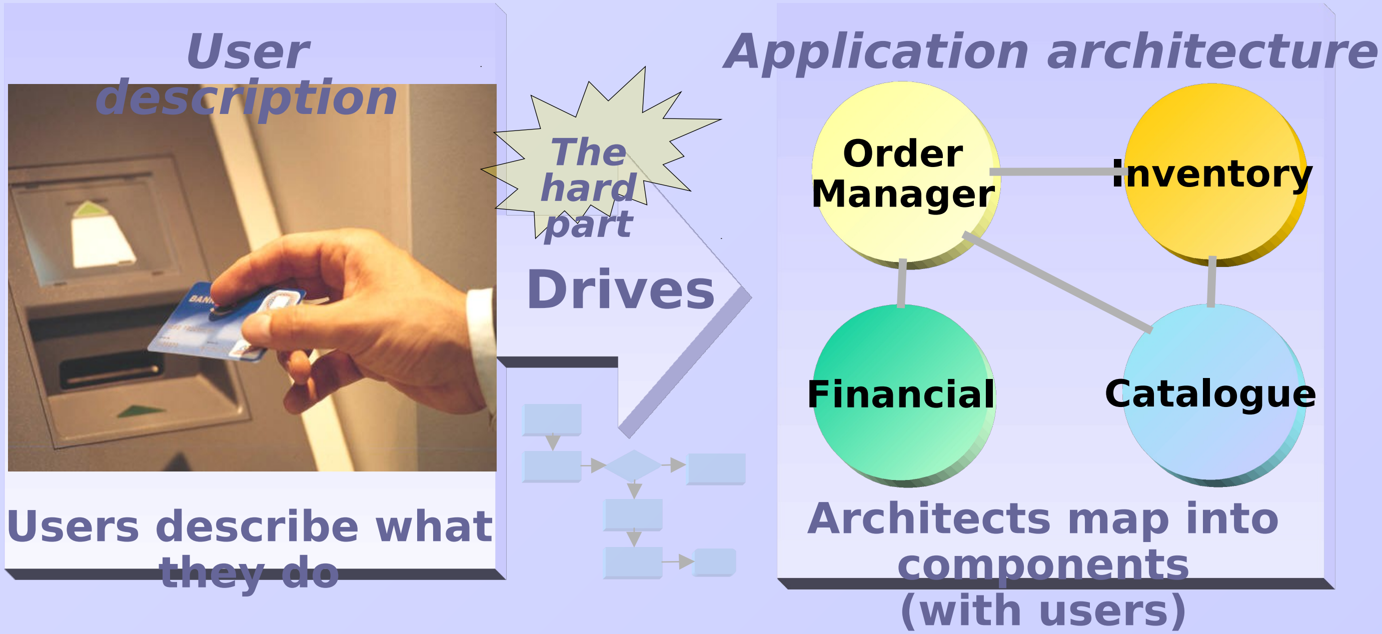
- SceneManagement
- Sensor
- Target
- Vehicle
- Terrain
- Atmosphere

How do you get components?

Use Case Modeling

Examples

- Withdraw $$
- Order item
- Fly thru



User description

Users describe what they do

The hard part

Drives

Application architecture

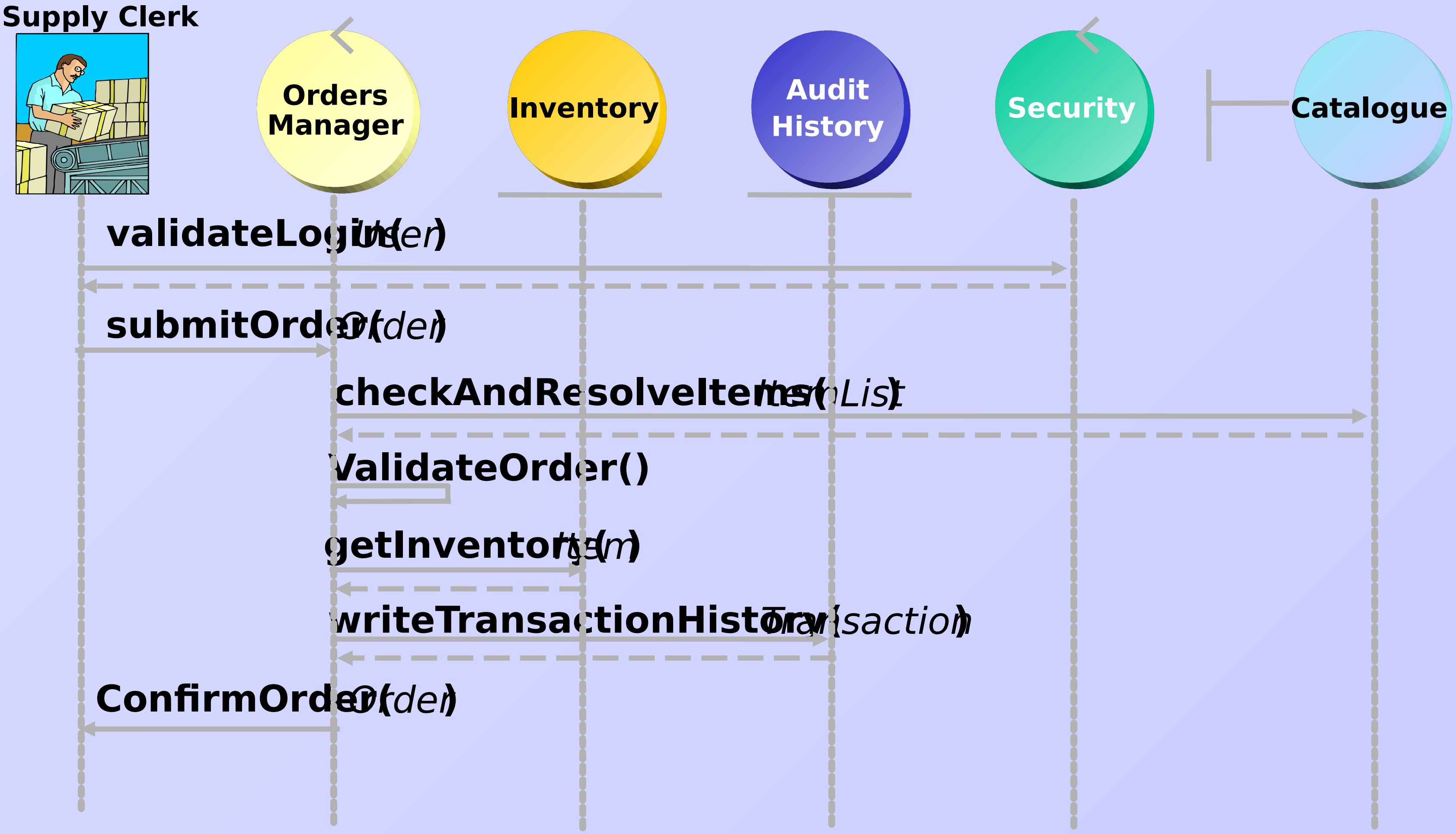
Architects map into components (with users)

- **Use case modeling**
  - Text, activity diagrams
  - User-centric
- **Components and interactions**
  - Architects map into enterprise components

Mission-focused

How components work together

(Order Item Use Case)



Supply Clerk

Orders Manager

Inventory

Audit History

Security

Catalogue

validateLogin(*User***)**

submitOrder(*Order***)**

checkAndResolveItems(*ItemList***)**

ValidateOrder()

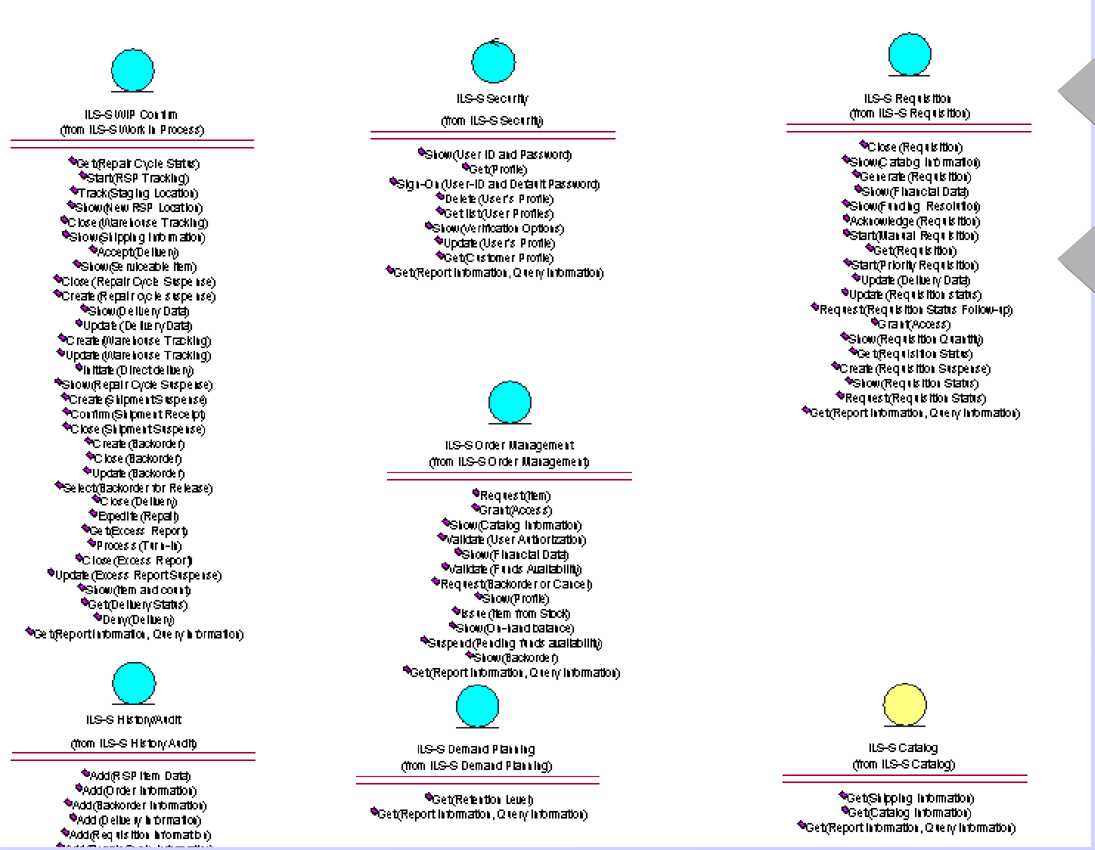
getInventory(*Item***)**

writeTransactionHistory(*Transaction***)**

ConfirmOrder(*Order***)**

Use Case Modeling Results

Suite of components with Interfaces


ILS-S WIP Confirm
(from ILS-S Work In Process)
Get(Repair Cycle Status)
Start(RSP Tracking)
Track(Staging Location)
Show(New RSP Location)
Close (Warehouse Tracking)
Show(Shipping Information)
Accept(Delivery)
Show(Serviceable Item)
Close (Repair Cycle Suspense)
Create (Repair cycle suspense)
Show(Delivery Data)
Update (Delivery Data)
Create (Warehouse Tracking)
Update (Warehouse Tracking)
Initiate (Direct delivery)
Show(Repair Cycle Suspense)
Create (Shipment Suspense)
Confirm (Shipment Receipt)
Close (Shipment Suspense)
Create (Backorder)
Close (Backorder)
Update (Backorder)
Select(Backorder for Release)
Close (Delivery)
Expedite (Repair)
Get(Excess Report)
Process (Turn-In)
Close (Excess Report)
Update (Excess Report Suspense)
Show(Item and count)
Get(Delivery Status)
Deny(Delivery)
Get(Report Information, Query Information)
ILS-S History/Audit
(from ILS-S History/Audit)
Add(RSP Item Data)
Add(Order Information)
Add(Backorder Information)
Add (Delivery Information)
Add(Requisition Information)
ILS-S Security
(from ILS-S Security)
Show(User ID and Password)
Get(Profile)
Sign-On (User-ID and Default Password)
Delete (User's Profile)
Get list(User Profiles)
Show(Verification Options)
Update (User's Profile)
Get(Customer Profile)
Get(Report Information, Query Information)
ILS-S Order Management
(from ILS-S Order Management)
Request(Item)
Grant(Access)
Show(Catalog Information)
Validate (User Authorization)
Show(Financial Data)
Validate (Funds Availability)
Request(Backorder or Cancel)
Show(Profile)
Issue (Item from Stock)
Show(On-hand balance)
Suspend(Pending funds availability)
Show(Backorder)
Get(Report Information, Query Information)
ILS-S Demand Planning
(from ILS-S Demand Planning)
Get(Retention Level)
Get(Report Information, Query Information)
ILS-S Requisition
(from ILS-S Requisition)
Close (Requisition)
Show(Catalog Information)
Generate (Requisition)
Show(Financial Data)
Show(Funding Resolution)
Acknowledge (Requisition)
Start(Manual Requisition)
Get(Requisition)
Start(Priority Requisition)
Update (Delivery Data)
Update (Requisition status)
Request(Requisition Status Follow-up)
Grant(Access)
Show(Requisition Quantity)
Get(Requisition Status)
Create (Requisition Suspense)
Show(Requisition Status)
Request(Requisition Status)
Get(Report Information, Query Information)
ILS-S Catalog
(from ILS-S Catalog)
Get(Shipping Information)
Get(Catalog Information)
Get(Report Information, Query Information)

Components

Interfaces

- **Interfaces define expected component behavior**

Components and Interfaces

Interfaces Define Component Behavior

- **Plug and play architecture**
  - Swap “approved” components in and out of scenario
  - Supports multiple modeling and visualization methods
    - Dynamic multi-scale modeling
    - Tunable rendering times
  - COTS insertion/interfaces
- **Self-assembling, capability-focused applications**
  - Components provide powerful toolkit
  - Use cases provide instruction manuals

Mission-focused, not platform centric

The Technical Approach

	Features	Benefits
User Centric	– Models actual system use – Use cases defined, refined – Workshop focused – Subject matter expert focused	– User knows exactly what they are getting – Application supports user needs – Capabilities focused, not technology driven
Architecture Centric	– Integrated with use cases – Design first, then develop	– Ensures application supports use cases – Ensures plug-and-play ability – Effectively leverages technology
Iterative Development	– Evolving prototype – Incrementally build application – Focus on hard problems first – Constantly reevaluating design	– Try ideas, test, reevaluate – Incorporate user feedback early in cycle – Ensures early acceptance, minimizes risk – Manages changing scope/requirements

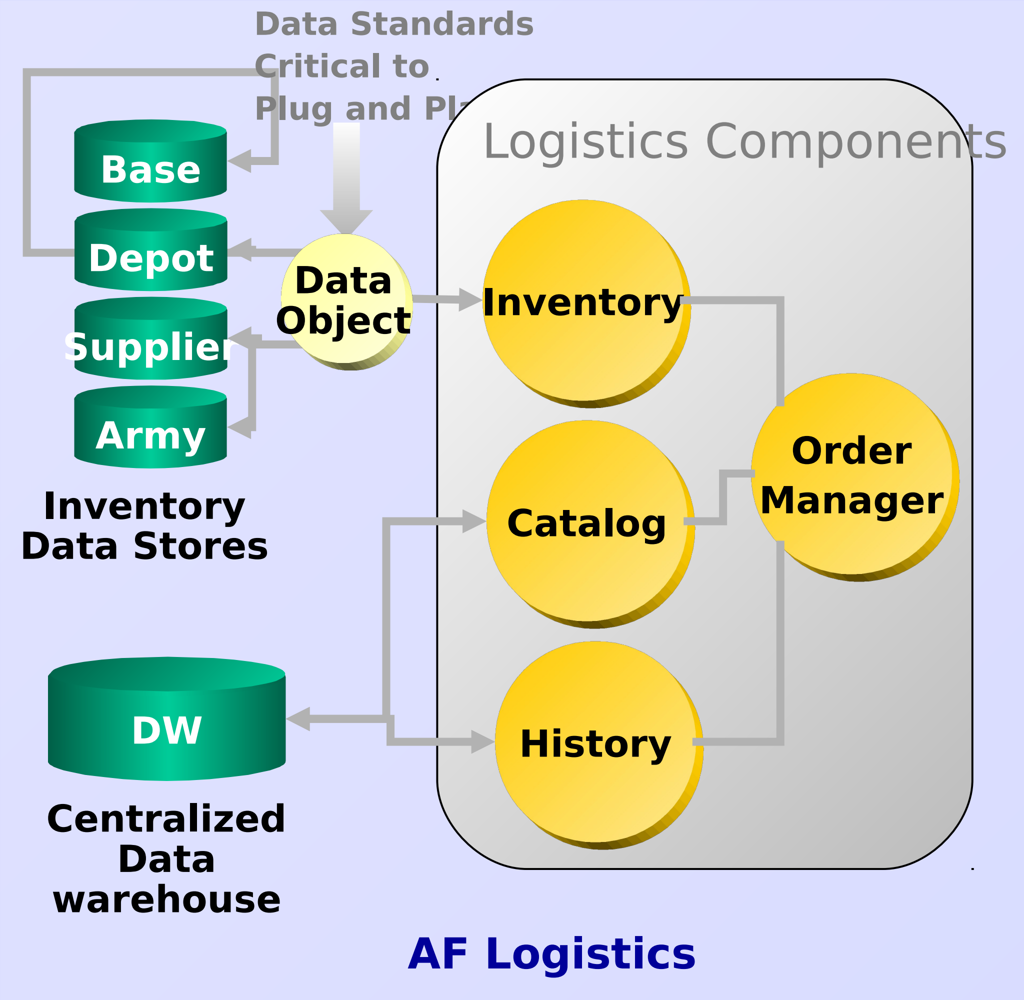
User Centric Development

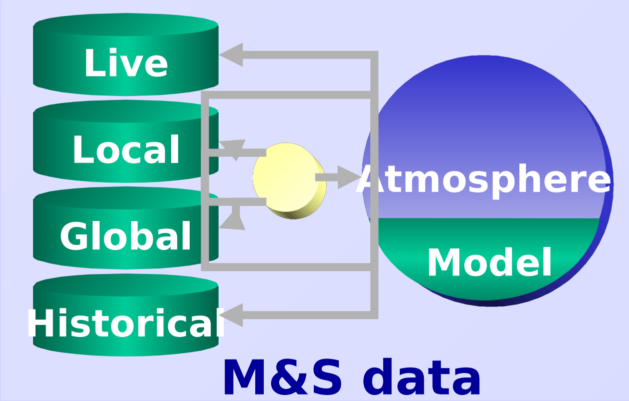
- **Use Case design**
  - Series of workshops with users to define and refine use cases

- Several groups provide ongoing support:
  - **Component User Groups (CUGs)**
    - Functional users define detailed component requirements
    - (Potentially need use-case-oriented Use Groups too)
  - **Software Component Working Groups (SCWGs)**
    - Review overarching architecture
    - Non-technical (more programmatic)
  - **Change Control Board**
    - Approval authority and funding for scope changes
    - Approves delivery and signs checks
  - **Architecture Oversight**
    - Oversees component architecture
    - Ensures functionality, reviews changes and overall design

But Where’s The Data?

Application interfaces separate from data sources





Data accessed through components

- CUGs define how components work
- Data designed around components
- Some enterprise-wide
  - Enterprise data warehouses
- Some local managed
  - Local operational data stores
  - Data volume
  - Security constraints

Challenges

- **Human factors**
  - Legacy people and legacy systems
    - Invite users to be part of change
  - Knowledge drain
    - SME and architects need to stay with projects
  - Cultural change

- **S&T**
  - DMSO needs better alignment with S&T community
    - Limited sphere of influence
    - Visualization versus science focus

- **Technology**
  - Issues more design-centric than technology centric
  - Component technology well-defined, but frameworks are immature
    - Emerging technologies: web services to intelligent agents to self-organizing networks

Funding Challenges

- **Funding (the BIGGEST challenge)**
  - DoD funds systems, not integrated environments
  - Initial development costs significantly higher
  - *Joint Synthetic Battlespace*
  - *Joint Virtual Battlespace*
    - Architecture driven by specific systems, not overarching architecture
  - *Air Force Logistics*

But, DoD is primed for transformation

Resistance is futile

Challenges

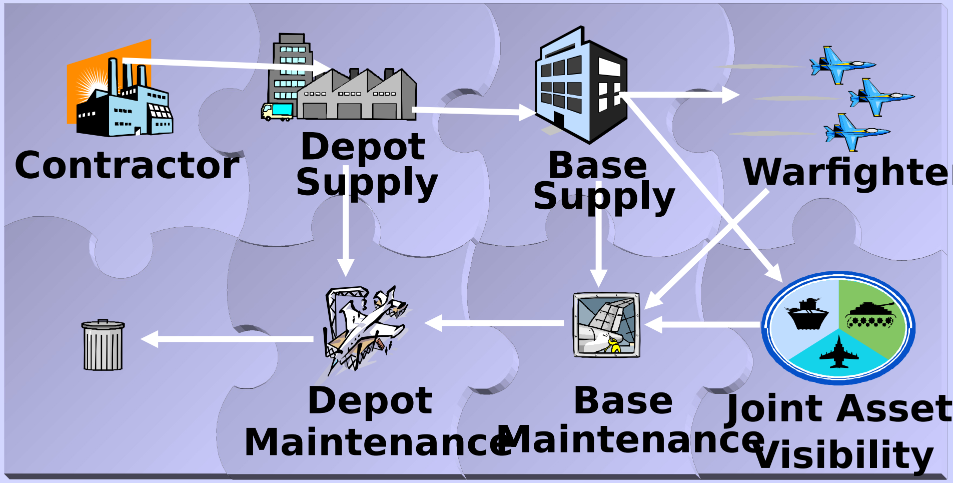
Begin shift of M&S from craft to scientific/engineering discipline

- Why is it an art craft?
  - Each system does it differently
  - Many experts, no common approach
  - Architecture and design is inherently an art form

- How can it become a scientific/engineering discipline
  - Use a methodology (consistent approach)
  - Reuse existing knowledge and models
  - Plug-and-play architectures and self-assembling models
  - Architectural oversight and guidance to new projects

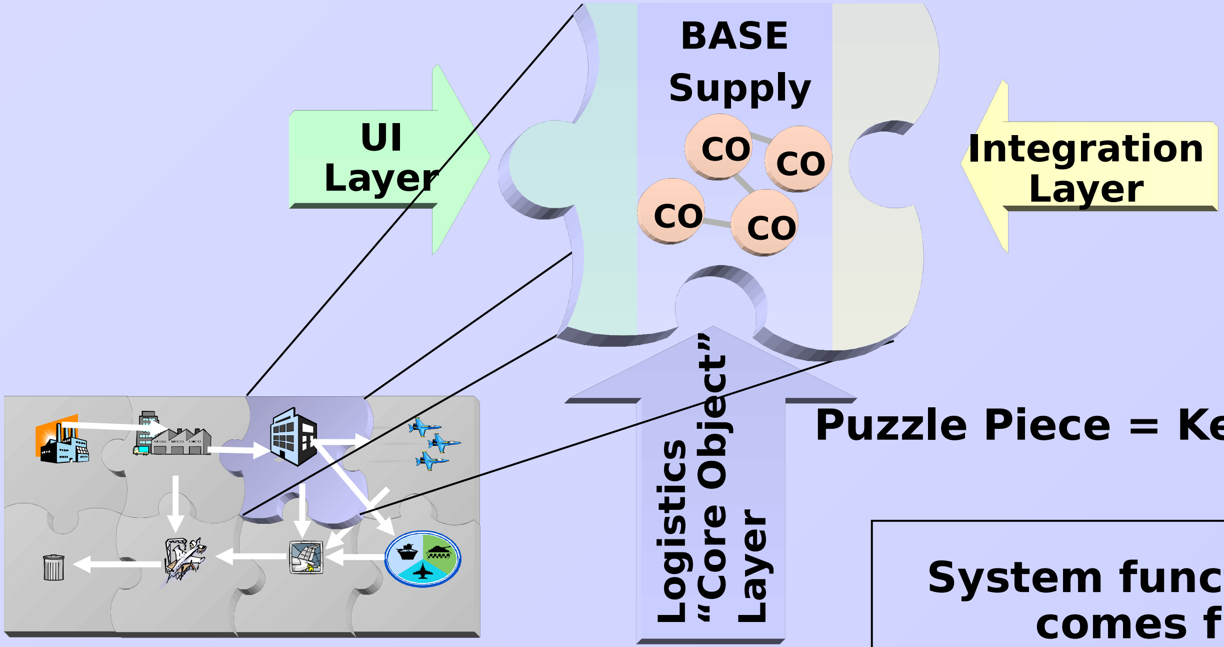
The rest of the talk focuses on addressing this challenge

Putting it all together



Integrated Air Force Logistics Vision

Putting it all together


UI Layer

BASE Supply

CO

CO

CO

CO

Integration Layer

Logistics "Core Object" Layer

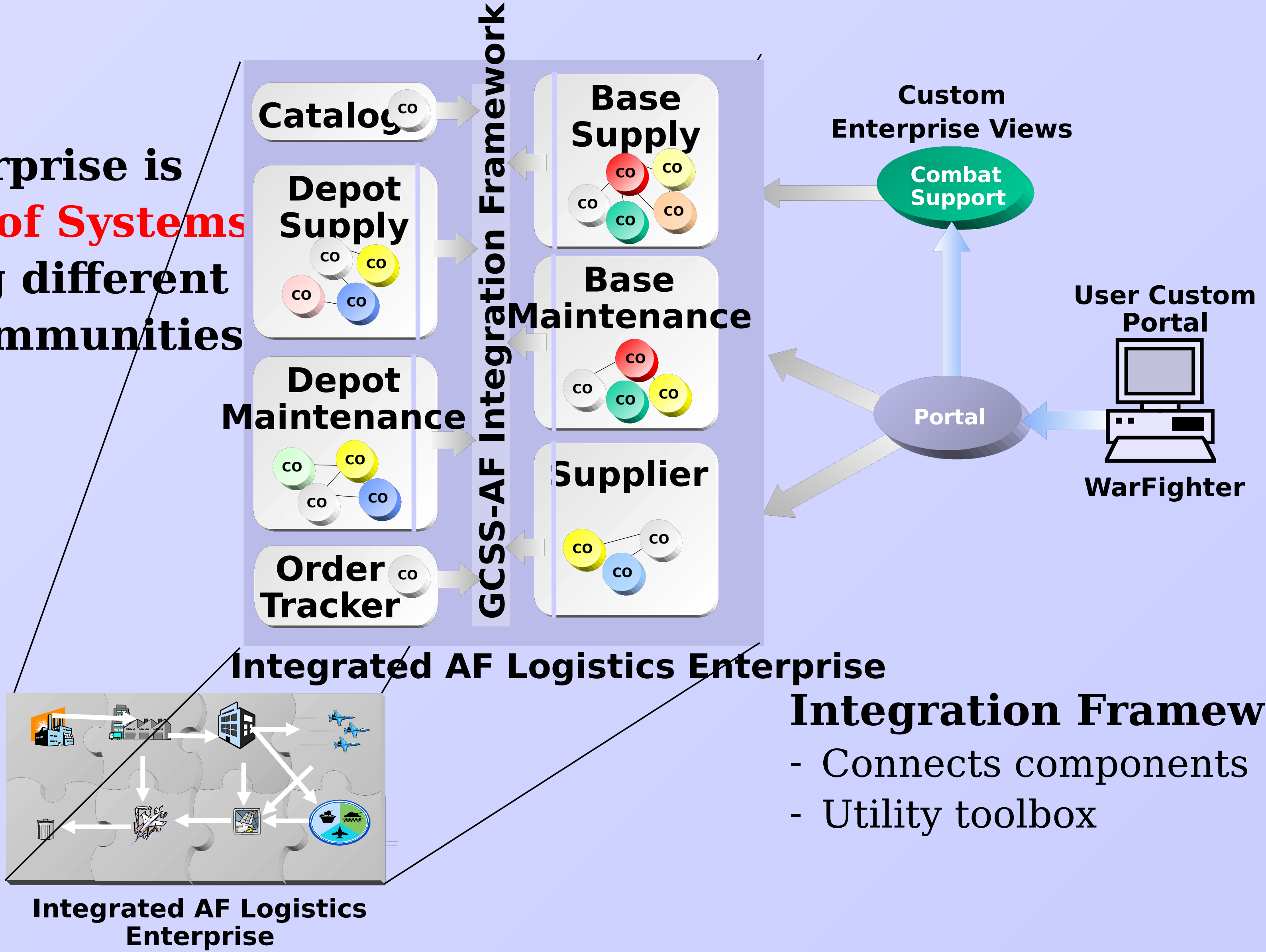
Puzzle Piece = Key Logistics *System*

Integrated AF Logistics Enterprise

System functionality comes from *Enterprise Components*

Putting it all together

Enterprise is System of Systems serving different user communities

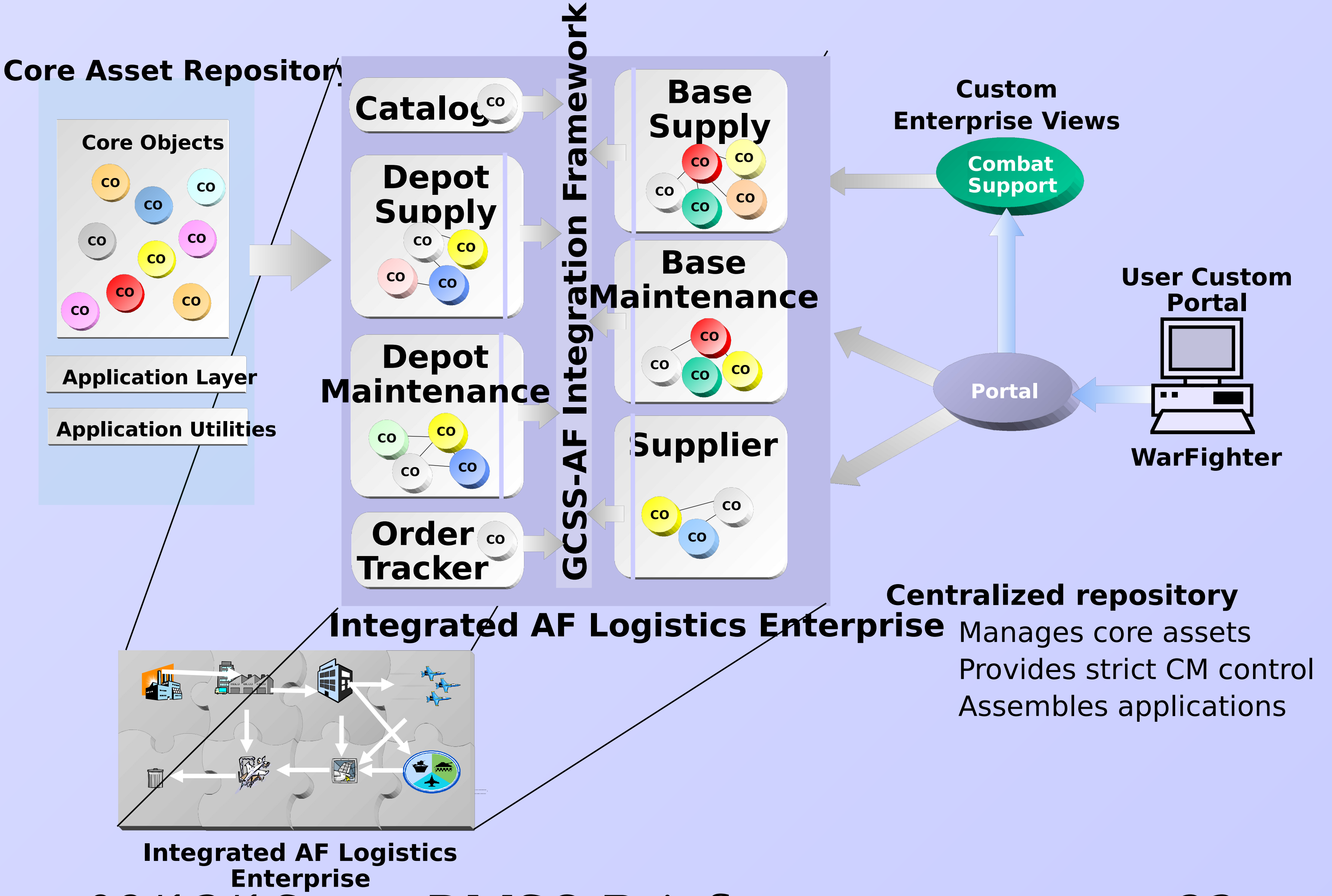


Integration Framework

- Connects components
- Utility toolbox

Putting it all together

Core Asset Repository



Centralized repository

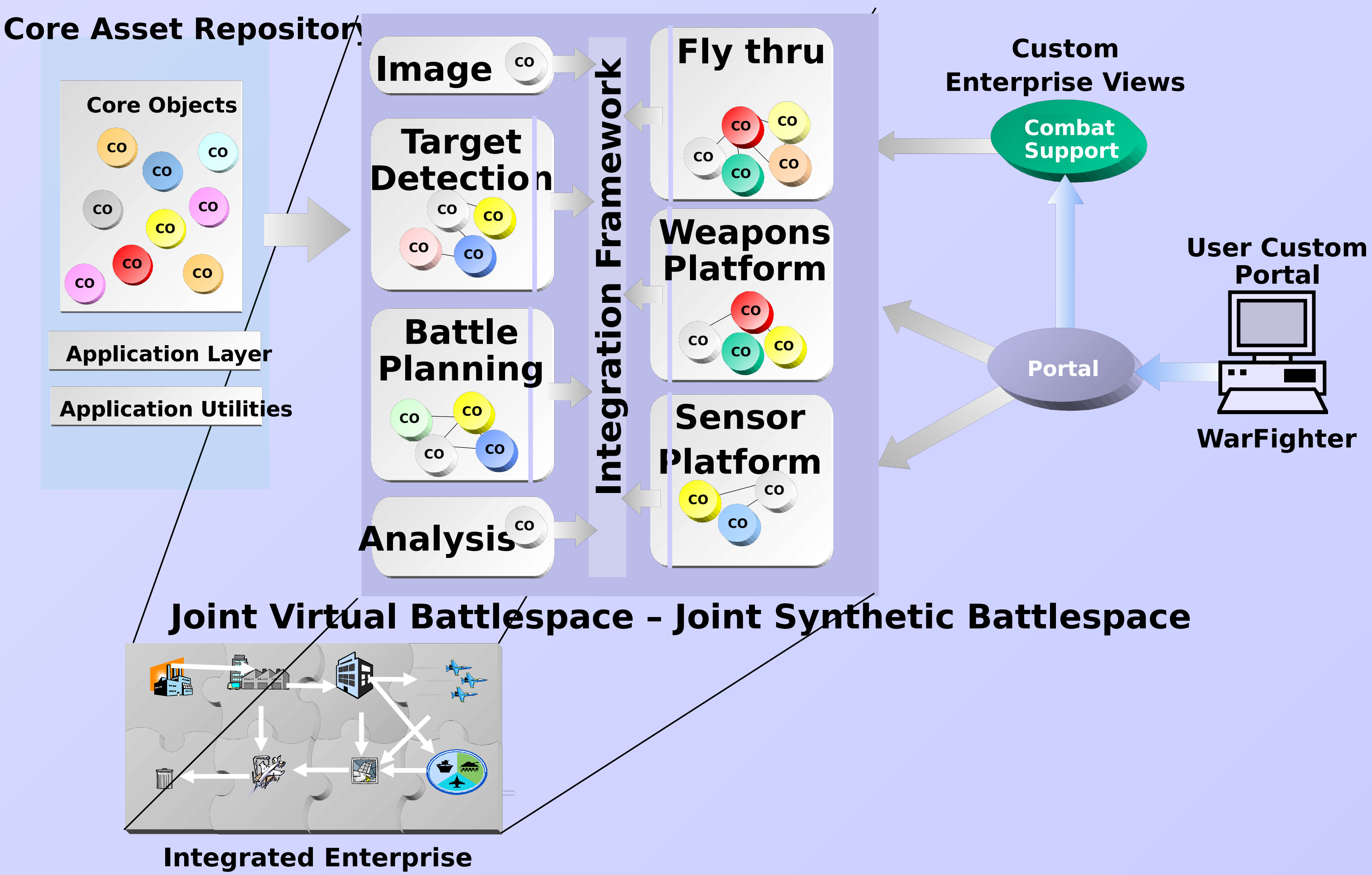
Manages core assets

Provides strict CM control

Assembles applications

Putting it all together

M&S Version



Core Asset Repository

Core Objects

CO

Application Layer

Application Utilities

Image

Target Detection

Battle Planning

Analysis

Integration Framework

Fly thru

Weapons Platform

Sensor Platform

Custom Enterprise Views

Combat Support

Portal

User Custom Portal

WarFighter

Joint Virtual Battlespace – Joint Synthetic Battlespace

Integrated Enterprise

Key Factors

- Components represent a different way of thinking
  - Both **enterprise** and **mission-centric**
  - Collection of parts designed to work together
  - Applications assembled around requirements, then disappear
- Not just technology
  - Technical approach very mission-focused
  - Driven by **users**, not technology
- Requires architecture oversight and expertise
  - Design and configuration management key pieces

Where does DMSO fit in?

- **Establish the methodology**
  - Use case modeling to component design
  - Don’t just advocate – teach, preach, and practice
- **Form SWAT teams to kickstart new projects**
  - Architectural oversight and process experts
  - Ensure they have the expertise to make it happen
- **Participate oversight committees**
  - CUGs, SCWGs, CCB, Architecture Oversight
  - Help each mission establish these as well
  - Maintain collective knowledge
- **Define the Component Repository**
  - Strict configuration management control
  - Certify components
  - Work with other groups on how to use it

SBA Observations (Revisited)

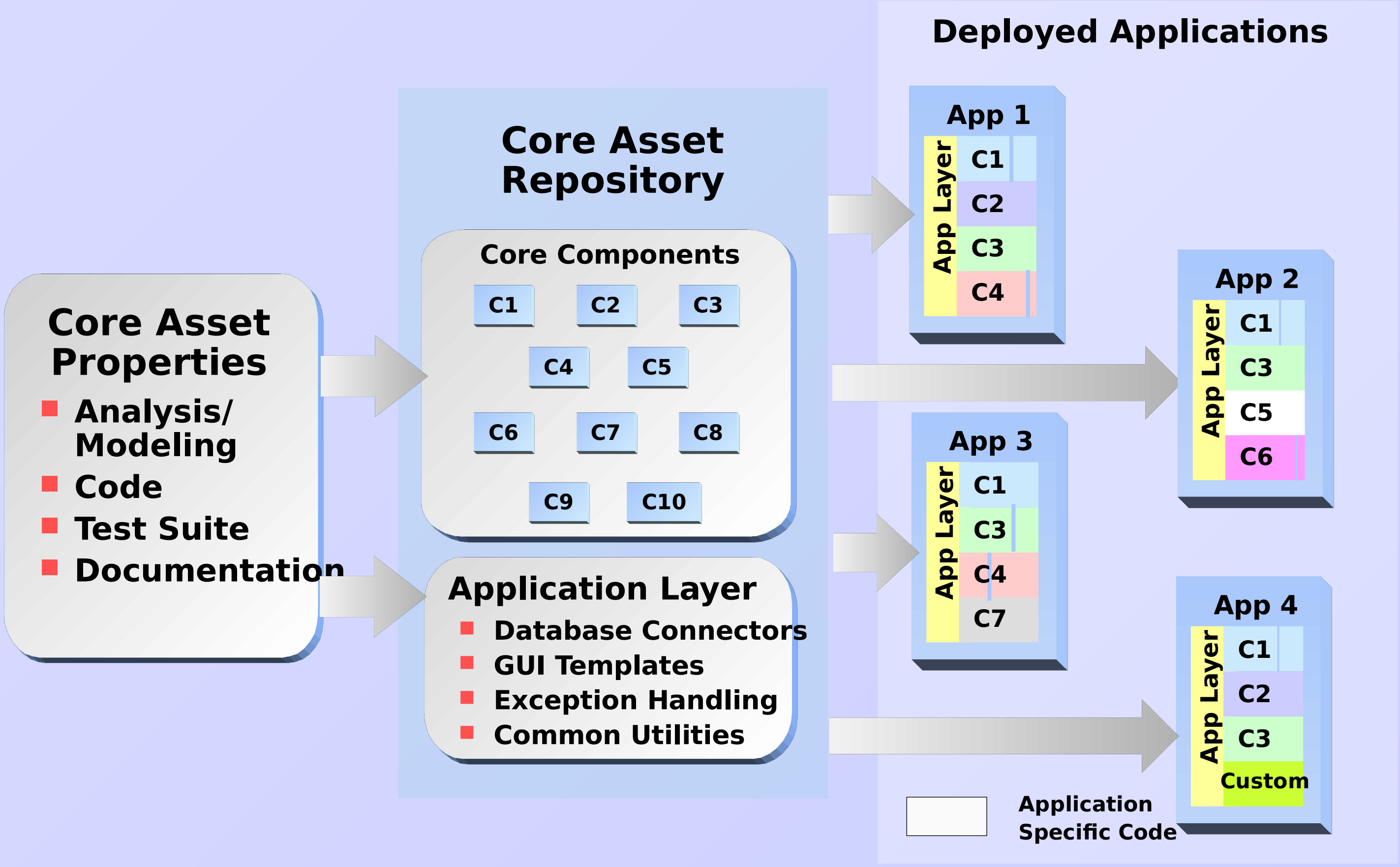
- Dr Sega – OSD, Director Defense Research & Engineering
  - Platform-centric→ network centric
  - Common vision representation
    - Multiple function areas
    - Joint, interoperable, re-useable models
- Dr Dahmann – Scientific advisor to Director of Interoperability
  - System focus → mission focus
- Ms. Zimmerman – DMSO
  - Rapidly composable and scalable M&S
  - Strong CM focus
  - Build only what is needed
- Mr Lunceford – Director, Army M&S Office
  - M&S best practices still a mystery
  - ***Begin shift of M&S from craft to scientific/engineering discipline***

What is a Software Product Line (SPL)?

- Software Product Line*
  - “a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.”

* From the book *Software Product Lines,* by Paul Clements and Linda Northrop

Software Product Line



Core Asset Properties

- Analysis/ Modeling
- Code
- Test Suite
- Documentation

Core Asset Repository

Core Components

C1 C2 C3
C4 C5
C6 C7 C8
C9 C10

Application Layer

- Database Connectors
- GUI Templates
- Exception Handling
- Common Utilities

Deployed Applications

App 1: App Layer, C1, C2, C3, C4

App 2: App Layer, C1, C3, C5, C6

App 3: App Layer, C1, C3, C4, C7

App 4: App Layer, C1, C2, C3, Custom

⬜ Application Specific Code

Why a SPL?

- SPL provides established methodology for reusable component development across multiple applications

- Core Asset Repository extends well beyond centralized code
  - Standardized requirements for all objects
    - Interface and functional
  - Complete test cases
  - Integrated with development/CM environment

- SPL provides mechanism for formal configuration management and testing components
  - Essential element to maintain plug-and-play capability
  - ***Ensures components always compatible with current architecture***

Next Steps

- Embrace component-based approach
- Setup workshops to define scenarios and approach
  - Focus on few key capabilities
    - Keep others in mind
  - Determine how existing pieces fit into this approach
- Identify initial project
- Begin with a demonstrable prototype

- Onward and upward!

Thank You

Me: Karl Garrison
Keane, Inc
1410 Spring Hill Road
McLean, VA

703-655-5620 (C)
703-848-7200 (O)
karl_c_garrison@keane.com